Constraint Programming Background and History

Jean-François Puget, IBM Distinguished Engineer, IBM
Twitter: @JFPuget
https://www.ibm.com/developerworks/mydeveloperworks/blogs/jfp/?lang=en

Paul Shaw, CP Products Development Lead, IBM

January 16, 2014

Disclaimer

- We work for IBM
  - The views expressed here are ours, not IBM's

- We worked for ILOG
  - Views expressed here may be biased towards ILOG / IBM past experience in this area
  - They may also biased towards IBM products in this area
    - IBM ILOG CP Optimizer
    - ILOG Solver

- But we think there is some general truth here

- Lots of CP research deals with programming language design, or theory of computing
  - We will not deal with this here

What Is Constraint Programming (CP) ?

Languages or systems for solving combinatorial (optimization) problems

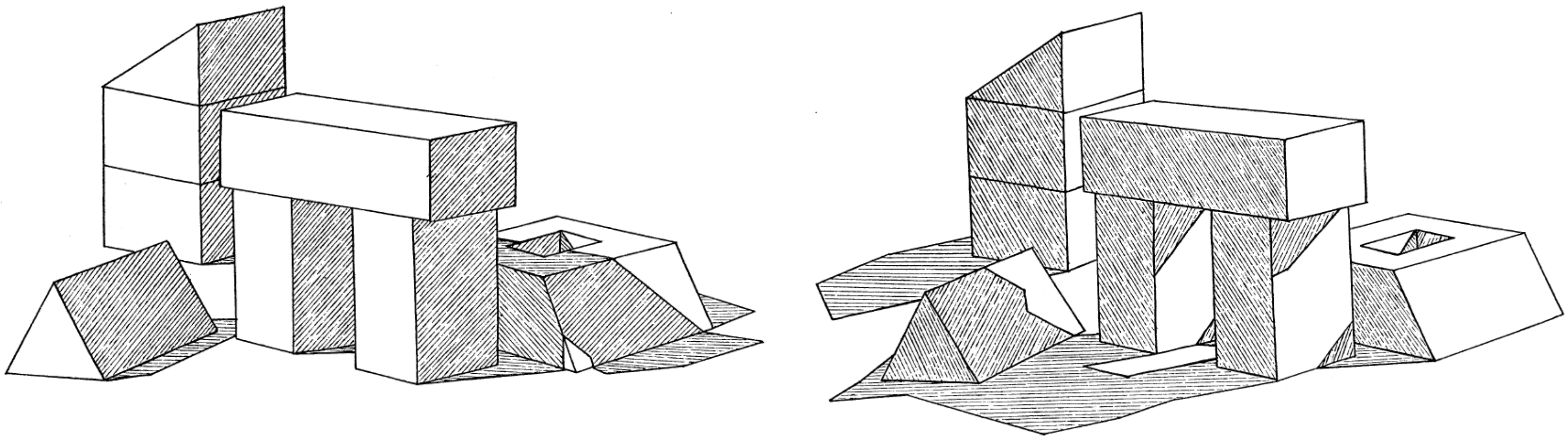
We will introduce major components of constraint programming via their appearance during the history of the field

CP draws on :

Artificial Intelligence, Computer Vision, Expert systems, Operations Research, Programming Languages Design, Mathematical Logic, Graph Theory, …

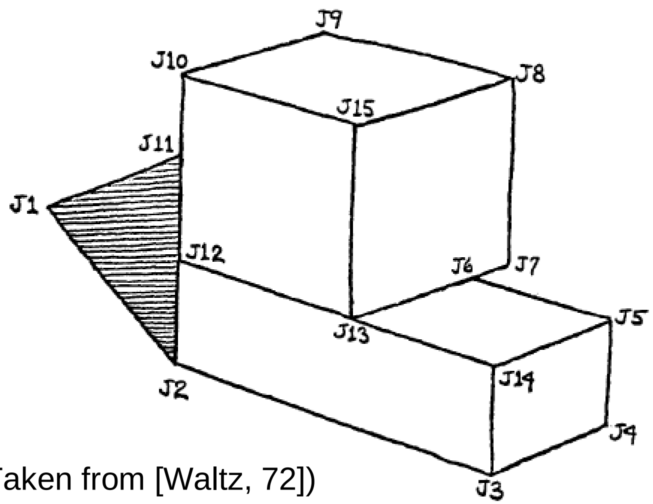
Waltz Filtering

Generating Semantic Descriptions From Drawings of Scenes With Shadows
David L. Waltz
1972

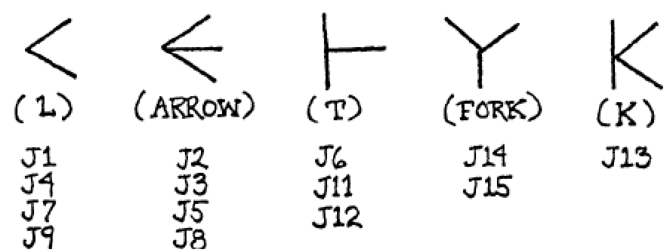


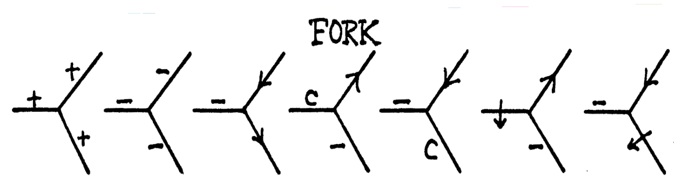
Waltz Filtering

- Edge labels
  - Convex, Concave, Shadow, Obscuring, Crack
- Junction types
  - L, Arrow, T, Fork, K
- Compute consistent edge labeling
  - Junctions have a finite number of admissible labeling of their edges
  - Edge label must be consistent over the junctions involving it
  - Method
    - Start with all possible edge labels
    - Remove labels that are inconsistent (filter)
    - Propagate the filtering to neighbouring edges of the image
    - Filter edge labels, continue propagation
    - Repeat the process until no more reductions are possible

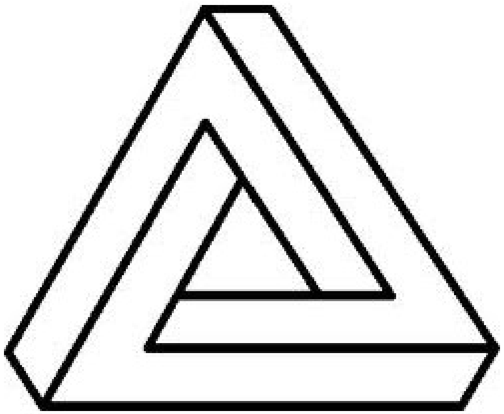


(Taken from [Waltz, 72])





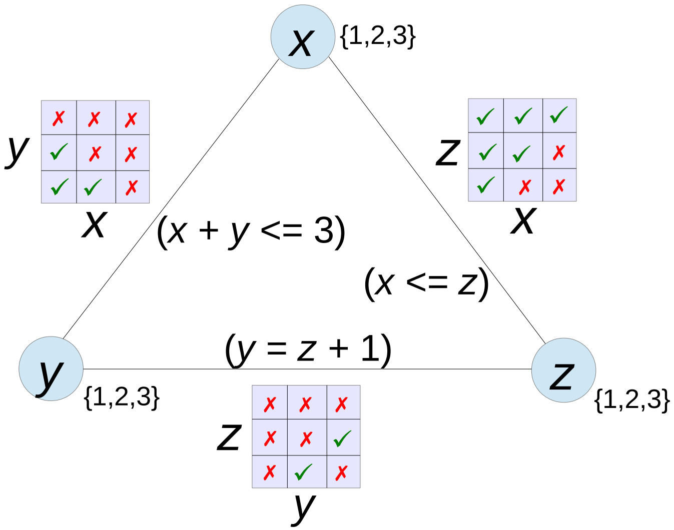
An Impossible Problem



- What happens if we apply Waltz filters to the Penrose triangle?
- The process "over-filters"
  - At least one edge has no possible label
  - That is, there is no solution
- Quite a nice result for such a simple idea!

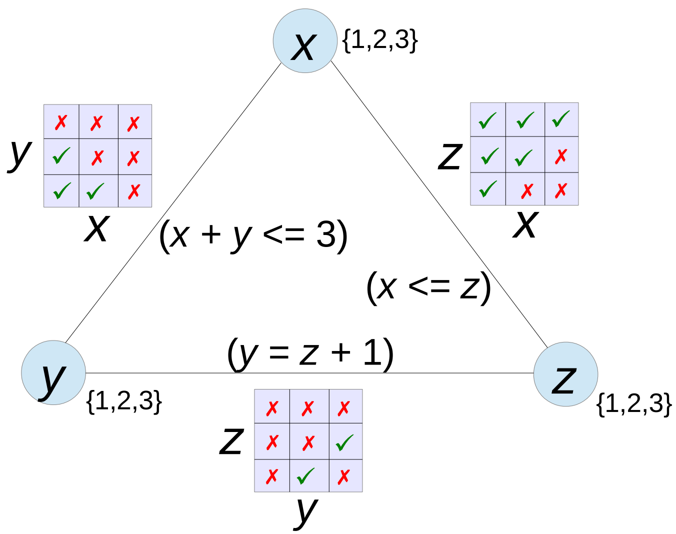
Constraint satisfaction problems (CSP)

- (Binary) Constraint Satisfaction problems were studied in the mid 1970s
- A CSP consists of variables, each with a *discrete domain* and binary constraints between pairs of variables
- A constraint is represented as a *relation* by simply listing the pairs of compatible values from each variable
  - Can also view a constraint as a value compatibility matrix



Constraint satisfaction problems (CSP)

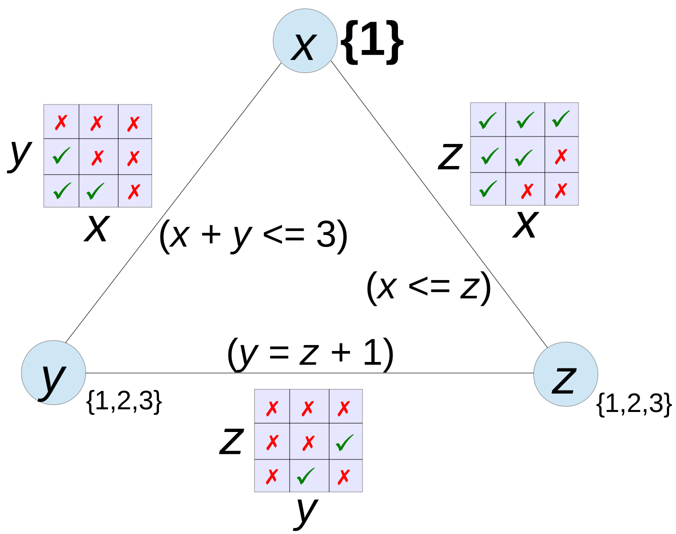
- (Binary) Constraint Satisfaction problems were studied in the mid 1970s
- A CSP consists of variables, each with a *discrete domain* and binary constraints between pairs of variables
- A constraint is represented as a *relation* by simply listing the pairs of compatible values from each variable
  - Can also view a constraint as a value compatibility matrix



- The first studies in the 1970s used *backward-oriented* tree search algorithms
- John Gaschnig studied various forms in the mid to late 1970s
  - backward checking
  - backmarking
  - backjumping

Constraint satisfaction problems (CSP)

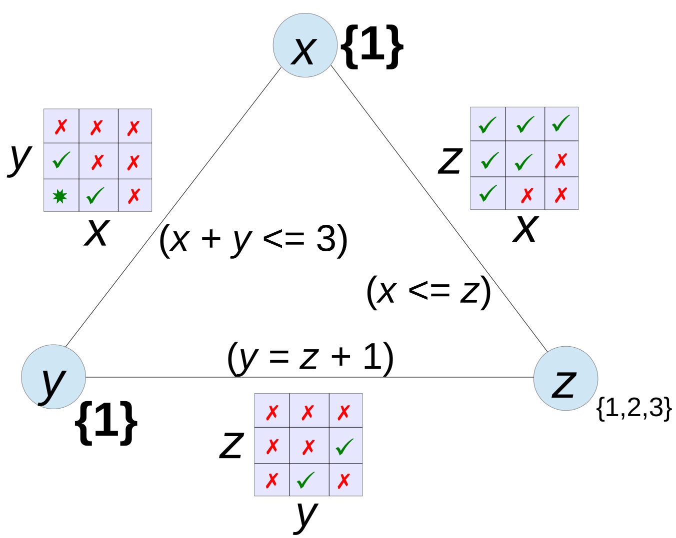
- (Binary) Constraint Satisfaction problems were studied in the mid 1970s
- A CSP consists of variables, each with a *discrete domain* and binary constraints between pairs of variables
- A constraint is represented as a *relation* by simply listing the pairs of compatible values from each variable
  - Can also view a constraint as a value compatibility matrix



- The first studies in the 1970s used *backward-oriented* tree search algorithms
- John Gaschnig studied various forms in the mid to late 1970s
  - backward checking
  - backmarking
  - backjumping

Constraint satisfaction problems (CSP)

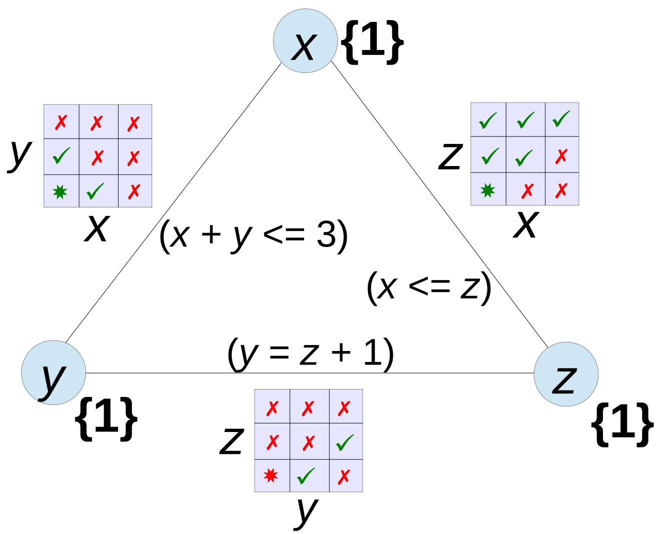
- (Binary) Constraint Satisfaction problems were studied in the mid 1970s
- A CSP consists of variables, each with a *discrete domain* and binary constraints between pairs of variables
- A constraint is represented as a *relation* by simply listing the pairs of compatible values from each variable
  - Can also view a constraint as a value compatibility matrix



- The first studies in the 1970s used *backward-oriented* tree search algorithms
- John Gaschnig studied various forms in the mid to late 1970s
  - backward checking
  - backmarking
  - backjumping

Constraint satisfaction problems (CSP)

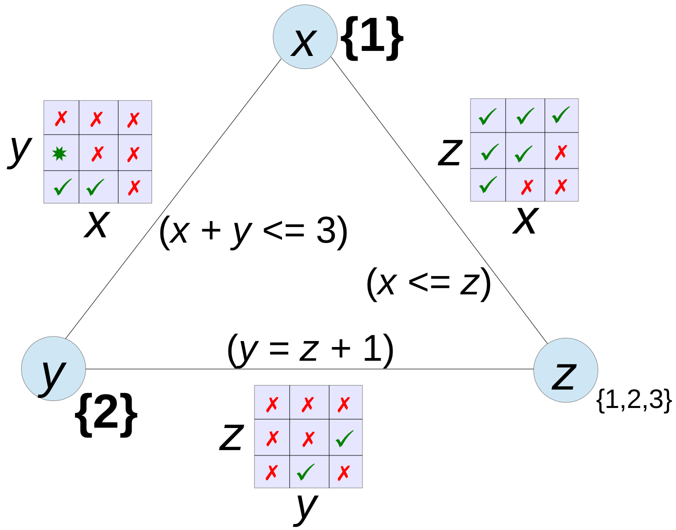
- (Binary) Constraint Satisfaction problems were studied in the mid 1970s
- A CSP consists of variables, each with a *discrete domain* and binary constraints between pairs of variables
- A constraint is represented as a *relation* by simply listing the pairs of compatible values from each variable
  - Can also view a constraint as a value compatibility matrix



- The first studies in the 1970s used *backward-oriented* tree search algorithms
- John Gaschnig studied various forms in the mid to late 1970s
  - backward checking
  - backmarking
  - backjumping

Constraint satisfaction problems (CSP)

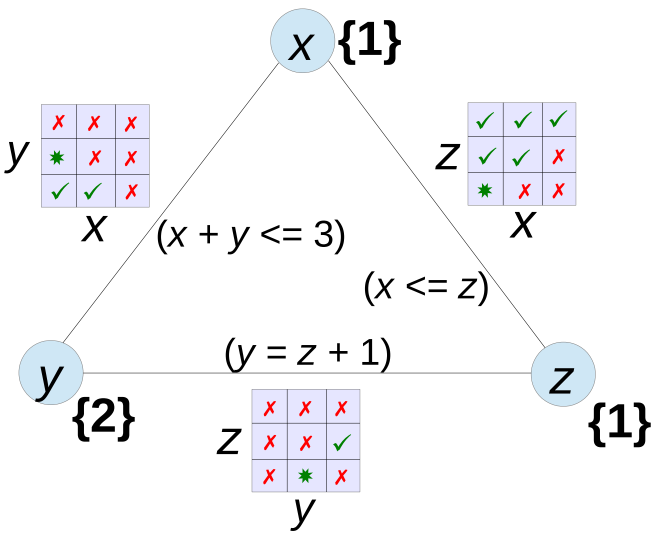
- (Binary) Constraint Satisfaction problems were studied in the mid 1970s
- A CSP consists of variables, each with a *discrete domain* and binary constraints between pairs of variables
- A constraint is represented as a *relation* by simply listing the pairs of compatible values from each variable
  - Can also view a constraint as a value compatibility matrix



- The first studies in the 1970s used *backward-oriented* tree search algorithms
- John Gaschnig studied various forms in the mid to late 1970s
  - backward checking
  - backmarking
  - backjumping

Constraint satisfaction problems (CSP)

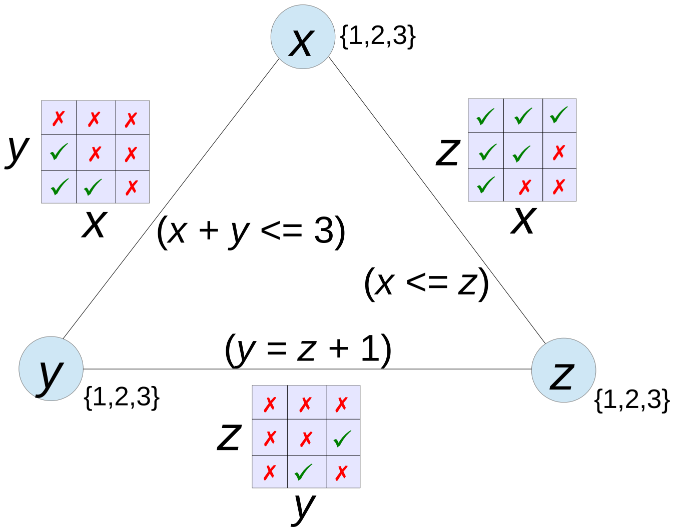
- (Binary) Constraint Satisfaction problems were studied in the mid 1970s
- A CSP consists of variables, each with a *discrete domain* and binary constraints between pairs of variables
- A constraint is represented as a *relation* by simply listing the pairs of compatible values from each variable
  - Can also view a constraint as a value compatibility matrix



- The first studies in the 1970s used *backward-oriented* tree search algorithms
- John Gaschnig studied various forms in the mid to late 1970s
  - backward checking
  - backmarking
  - backjumping

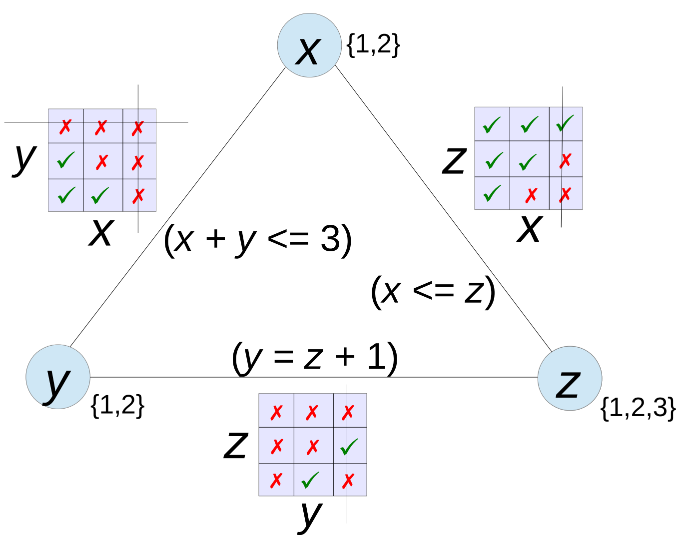
Later developments

- Generalization of Waltz filtering
  - Resulted in the “arc-consistency” algorithm of Mackworth (1977)
- For a given CSP, Mackworth's algorithm filters the domains maximally, as viewed by each constraint
  - For a constraint on *x* and *y*, and for each possible value of *x*, a compatible value of *y* is available (and vice versa)



Later developments

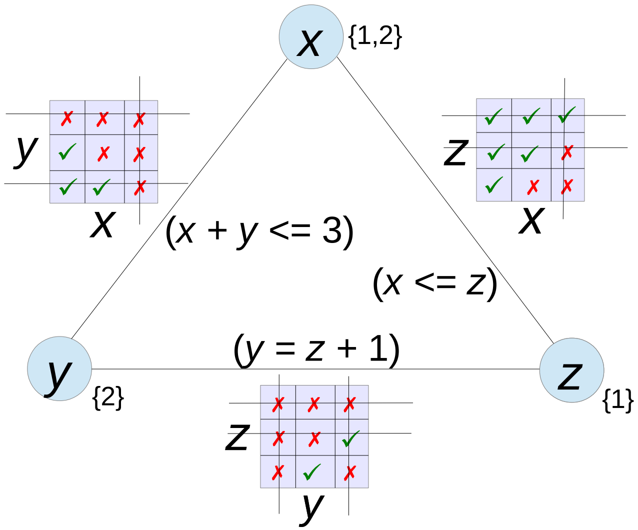
- Generalization of Waltz filtering
  – Resulted in the “arc-consistency” algorithm of Mackworth (1977)
- For a given CSP, Mackworth's algorithm filters the domains maximally, as viewed by each constraint
  – For a constraint on *x* and *y*, and for each possible value of *x*, a compatible value of *y* is available (and vice versa)



- Looking at constraint *“x-y”*
  – *x*=3 is not possible
  – *y*=3 is not possible

Later developments

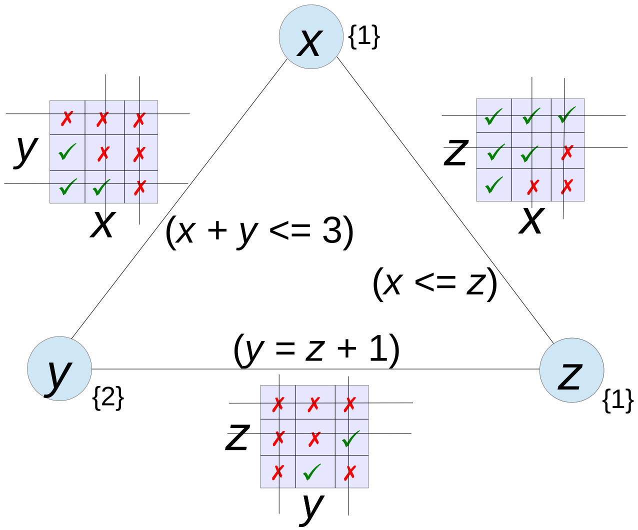
- Generalization of Waltz filtering
  - Resulted in the "arc-consistency" algorithm of Mackworth (1977)
- For a given CSP, Mackworth's algorithm filters the domains maximally, as viewed by each constraint
  - For a constraint on *x* and *y*, and for each possible value of *x*, a compatible value of *y* is available (and vice versa)



- Looking at constraint "*x*-*y*"
  - *x*=3 is not possible
  - *y*=3 is not possible
- Looking at constraint "*y*-*z*"
  - *y*=1 is not possible
  - *z*=2 is not possible
  - *z*=3 is not possible

Later developments

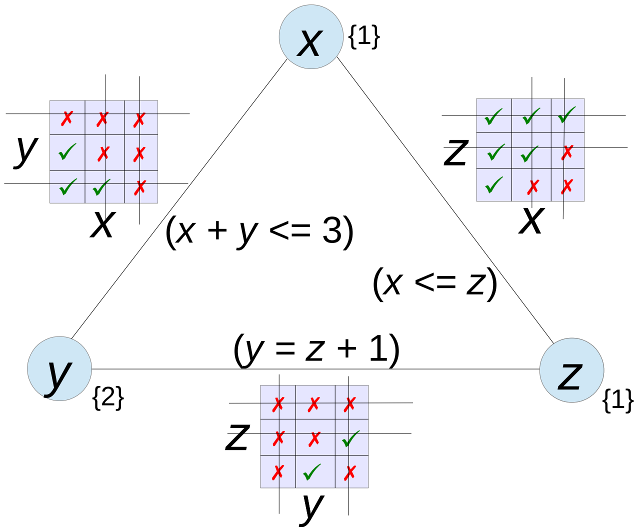
- Generalization of Waltz filtering
  - Resulted in the "arc-consistency" algorithm of Mackworth (1977)
- For a given CSP, Mackworth's algorithm filters the domains maximally, as viewed by each constraint.
  - For a constraint on *x* and *y*, and for each possible value of *x*, a compatible value of *y* is available (and vice versa)



- Look at constraint *"x-z"*
  - *x*=2 is not possible
- Here, all variables are fixed. In typical problems, there will generally just be a domain reduction

Later developments

- Generalization of Waltz filtering
  - Resulted in the “arc-consistency” algorithm of Mackworth (1977)
- For a given CSP, Mackworth's algorithm filters the domains maximally, as viewed by each constraint.
  - For a constraint on *x* and *y*, and for each possible value of *x*, a compatible value of *y* is available (and vice versa)



- The fusion of a tree search backtracking algorithm with Mackworth's algorithm invoked at each node became the newest way to solve CSPs
- It is the basis of the techniques used in CP solvers today
- The “constraint-centic” vision is still the usual method

The AC-3 Algorithm

Q = { (j,k) : (j,k) in arcs(G) }
while Q not empty do
    select and delete any arc *(l,m)* from *Q*
    if revise*(l,m)* then
        add all arcs *(n,l)* to *Q* where *(n,l)* in arcs(*G*)

revise(*j,k*)
    del = false
    for all *x* in domain(*j*)
    if there is no *y* in domain(*k*) such that $C_{jk}(x,y)$
        delete *x* from domain(*j*)
        del = true
    return *del*

ALICE : A Language for an Intelligent Combinatorial Exploration (Jean-Louis Laurière, 1976)

- Generic system for solving combinatorial problems
- Main features
  - Problem stated as the search for functions betwen finite sets subject to some constraints
  - Constraints could be logical or algebraic
  - Had a modelling language resembling what we see today
  - Black box solving module
    - Using dynamic rules to determine how the solver should behave at each node is the search
    - Had a kind of “portfolio” of rules
    - Used bipartite graphs to represent a function internally
  - Could optimize an objective function and provide a proof
- Had a big influence in France where CP research is most active

Timetabling Example

```
SOIT constante      N, P    ; sessions number, slots number
        ensemble  S = [1 N] ; session set
                   H = [1 P]       ; slot set
TROUVER fonction  F : S -> H  DIS         DMA
  ; DIS : slot disjunction (exclusion)
  ; DMA : max degree on sessions (number of rooms)
AVEC MIN     MAX     F(i)
                     i dans S
     F(5) < F(10)
     F(11) > F(4)   ET  F(11) > F(6)
FIN
  11, 10  ; values for N, P
  2, 3, 5, 7, 8, 10  ; sessions in disj. with 1… (1/2 matrix)
    3, 3, 3, 3, 3, 3, 3, 3, 3, 3  ; max degree for elements in H
 FIN
```

Constraint Logic Programming (mostly 1980s)

- The PROLOG language, (Colmerauer, 1973)
  - Backtracking search
  - Solve equalities between labeled trees
- Extended in the 1980s to handle more general constraints
  - Finite domains (CSP)
  - Linear programming
- A number of systems were created
  - Prolog III
  - CHIP
  - CLP(R)
  - Eclipse

An Eclipse example

```
    S E N D
+   M O R E
= M O N E Y
```

```
smm :-  X = [S,E,N,D,M,O,R,Y],
        X :: 0 .. 9,
        M #> 0,
        S #> 0,
        1000*S + 100*E + 10*N + D
        + 1000*M + 100*O + 10*R +
        E #= 10000*M + 1000*O +
        100*N + 10*E + Y,
        alldistinct(X),
        labeling(X),
        write(X).
```

Progress made in the 1980s

- Structure could be modelled
  - *e.g.* `alldistinct` in the previous Eclipse model
  - The “element” constraint or expression allows the user to index an array of values using a decision variable
  - The hope is that suitable solvers would be able to better solve structured models where semantics are better preserved
    - For example, one can do better domain reduction on more structured models
  - Better realized in the 90s via powerful *global* constraints
- Black box search, as done in ALICE was seen as too limited for real problems
  - Systems like CHIP allowed the user to program the search process

Constraint Programming Toolkits (late 80s until today)

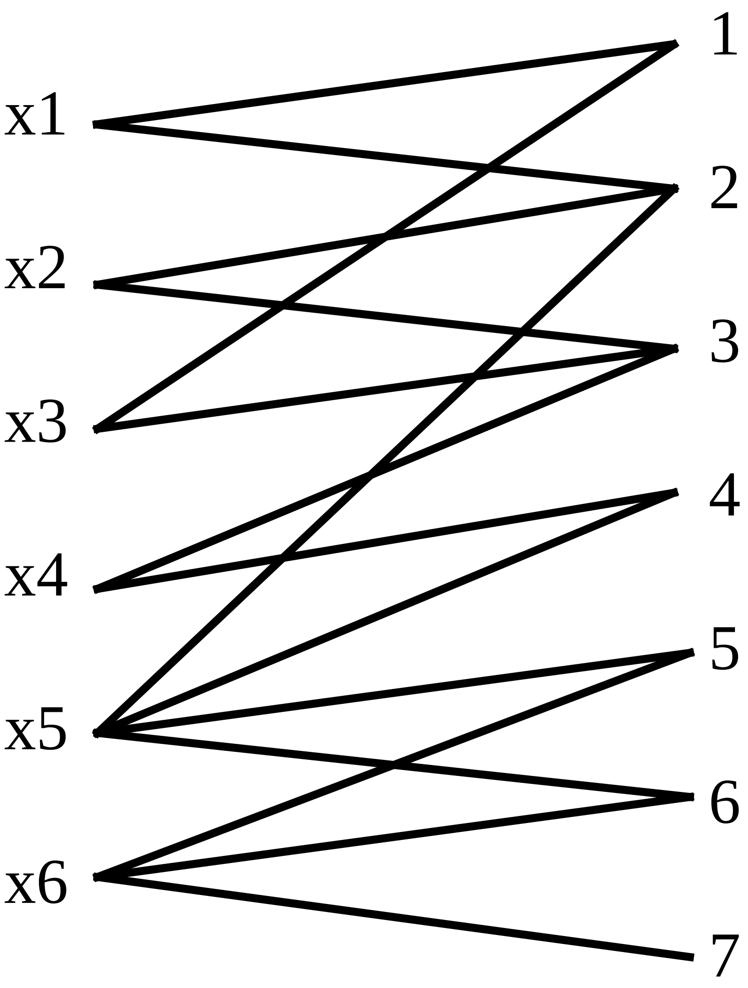
- Instead of designing a new language, implement a library in a host language. For example:
  - Lisp
    - PECOS (Puget, 1990)
  - C++
    - ILOG Solver (Puget, 1992)
    - Gecode (Schulte *et al.*, 2005)
    - OR-Tools (Perron *et al.*, 2009)
  - Java
    - CHOCO (Laburthe *et al.*)

- Decision variable types
  - Sets, task (scheduling), object, classes, strings
- Constraint types
  - Logical, arithmetic, set, *finite capacity resources*, graphs
  - Numerous *global constraints*
- Flexibility for implementing search procedures
- Hybridization with other domains
  - SAT
    - No-good learning
    - Propagation (watched literals)
    - Lazy clause generation
  - MP (SCIP, Achterberg)
  - Local search (LNS, Shaw)

Global Constraint Example: "All Different" Constraint (Regin 1994)

The value graph:

$x1 \in \{1,2\}$
$x2 \in \{2,3\}$
$x3 \in \{1,3\}$
$x4 \in \{3,4\}$
$x5 \in \{2,4,5,6\}$
$x6 \in \{5,6,7\}$



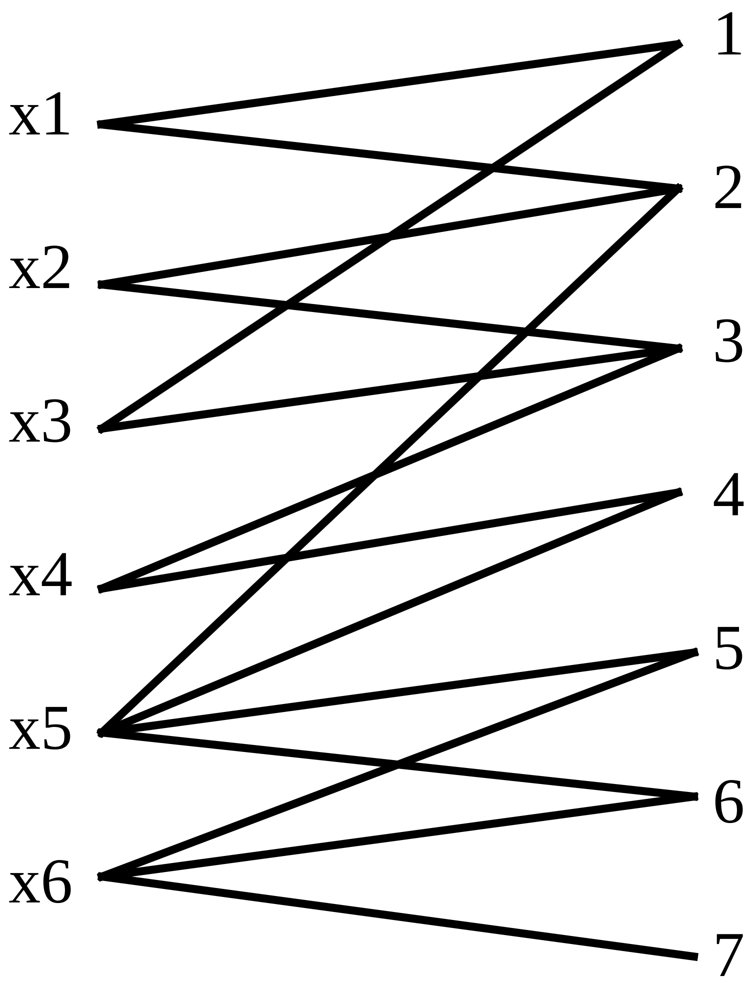
The bipartite graph represents the domains of the variables.

An arc from x2 to 3 means that 3 is a possible domain value for x2

Global Constraint Example: "All Different" Constraint (Regin 1994)

The value graph:

$x1 \in \{1,2\}$
$x2 \in \{2,3\}$
$x3 \in \{1,3\}$
$x4 \in \{3,4\}$
$x5 \in \{2,4,5,6\}$
$x6 \in \{5,6,7\}$



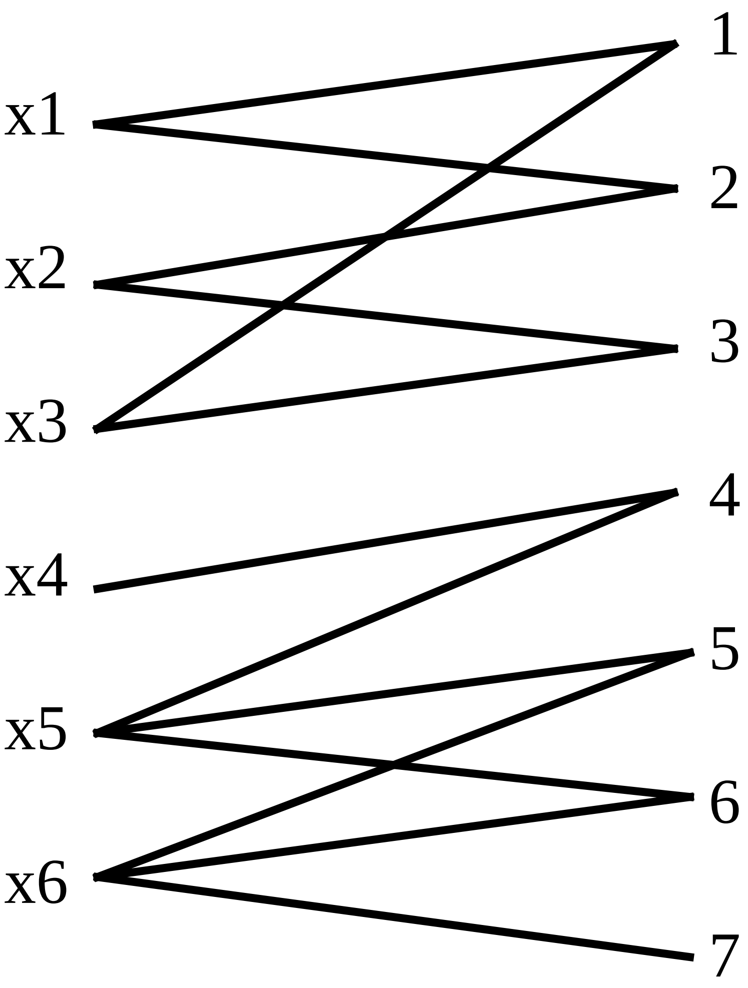
Between them, variables x1, x2 and x3 can take only values in {1,2,3}

Since three variables cover 3 values and all variables must take different values, no other variable can take values in {1,2,3}

Global Constraint Example: “All Different” Constraint (Regin 1994)

The value graph:

$x1 \in \{1,2\}$
$x2 \in \{2,3\}$
$x3 \in \{1,3\}$
$x4 \in \{4\}$
$x5 \in \{4,5,6\}$
$x6 \in \{5,6,7\}$

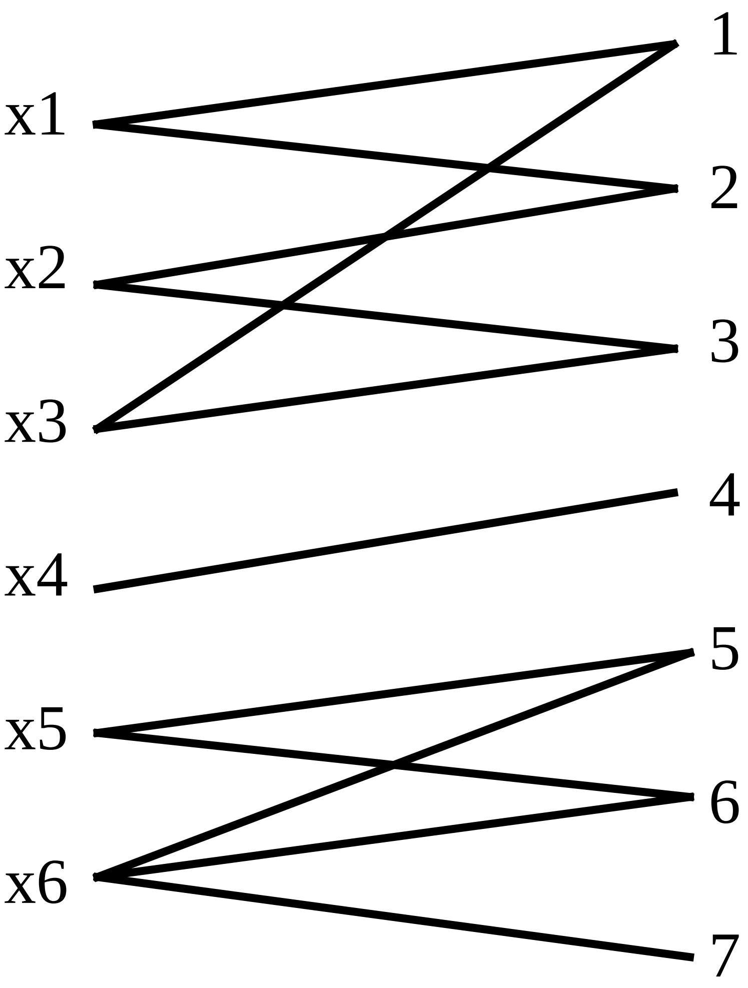


x4 must take the value 4 and so no other variable can take the value 4

Global Constraint Example: “All Different” Constraint (Regin 1994)

The value graph:

$x1 \in \{1,2\}$
$x2 \in \{2,3\}$
$x3 \in \{1,3\}$
$x4 \in \{4\}$
$x5 \in \{5,6\}$
$x6 \in \{5,6,7\}$



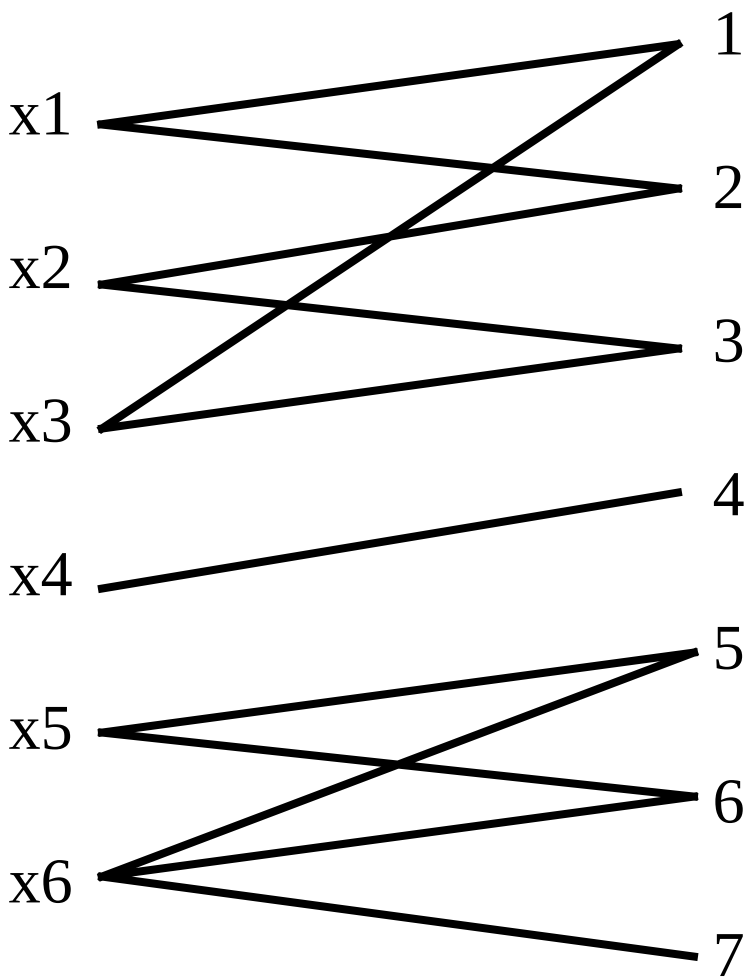
Regin's filtering algorithm is based on matching theory and identification of strongly connected components.

It was one of the first to use non-trivial algorithms to boost domain reduction

Global Constraint Example: “All Different” Constraint (Regin 1994)

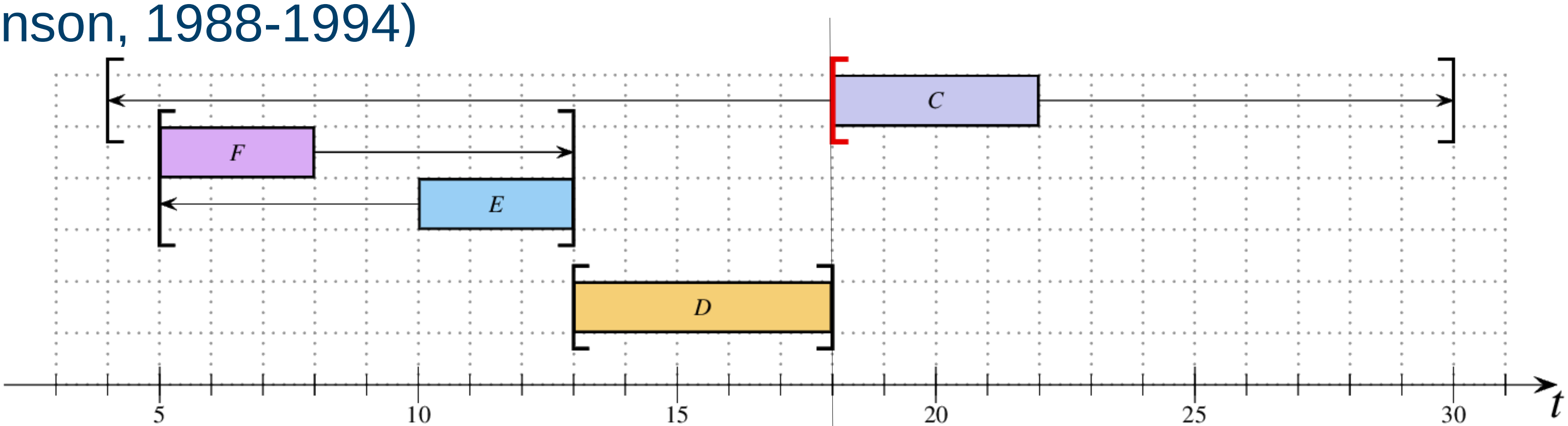
The value graph:

$x1 \in \{1,2\}$
$x2 \in \{2,3\}$
$x3 \in \{1,3\}$
$x4 \in \{4\}$
$x5 \in \{5,6\}$
$x6 \in \{5,6,7\}$



The “one specialized algorithm per constraint” method was pioneered in the 1980 and is now the standard method of buildinging CP Solvers in preference to methods like Macworth's AC-3

Global Constraint Example: Scheduling (Edge Finding, Carlier and Pinson, 1988-1994)



- Consider some deadline t.
- Θ = all activities that must finish before t.
- Λ = all activities that can start before t but can finish after t.
- If we can add one activity from Λ into Θ, how big earliest completion time we can make?
- Is it bigger than t?
- If yes, activity we used from Λ can be updated and removed from Λ.

- for example $t = lct_D = 18$
- Θ = {D, E, F}
- Λ = {C}
- $ECT_{\{C,D,E,F\}} = 19$
- Yes: 19 > 18
- $est_C := 18$

From Toolkits to Solvers

- ALICE was a black box
  - Users would state the problem in a declarative way (a model)
  - Similar to MP solvers like CPLEX
- CP progress was towards a “clear” box
  - Emphasis was maximum flexibility
    - Programmable search
    - Programmable constraint propagation
- Skill set needed to use CP was growing:
  - Large family of constraints types (100s)
  - In general, little effort on making simple things simple
  - Adoption was decreasing in the industry
  - MIP solvers were more and more adopted
- We tried to learn from MIP solvers

CP Next Challenge: Simplicity of Use
(Puget 2004)

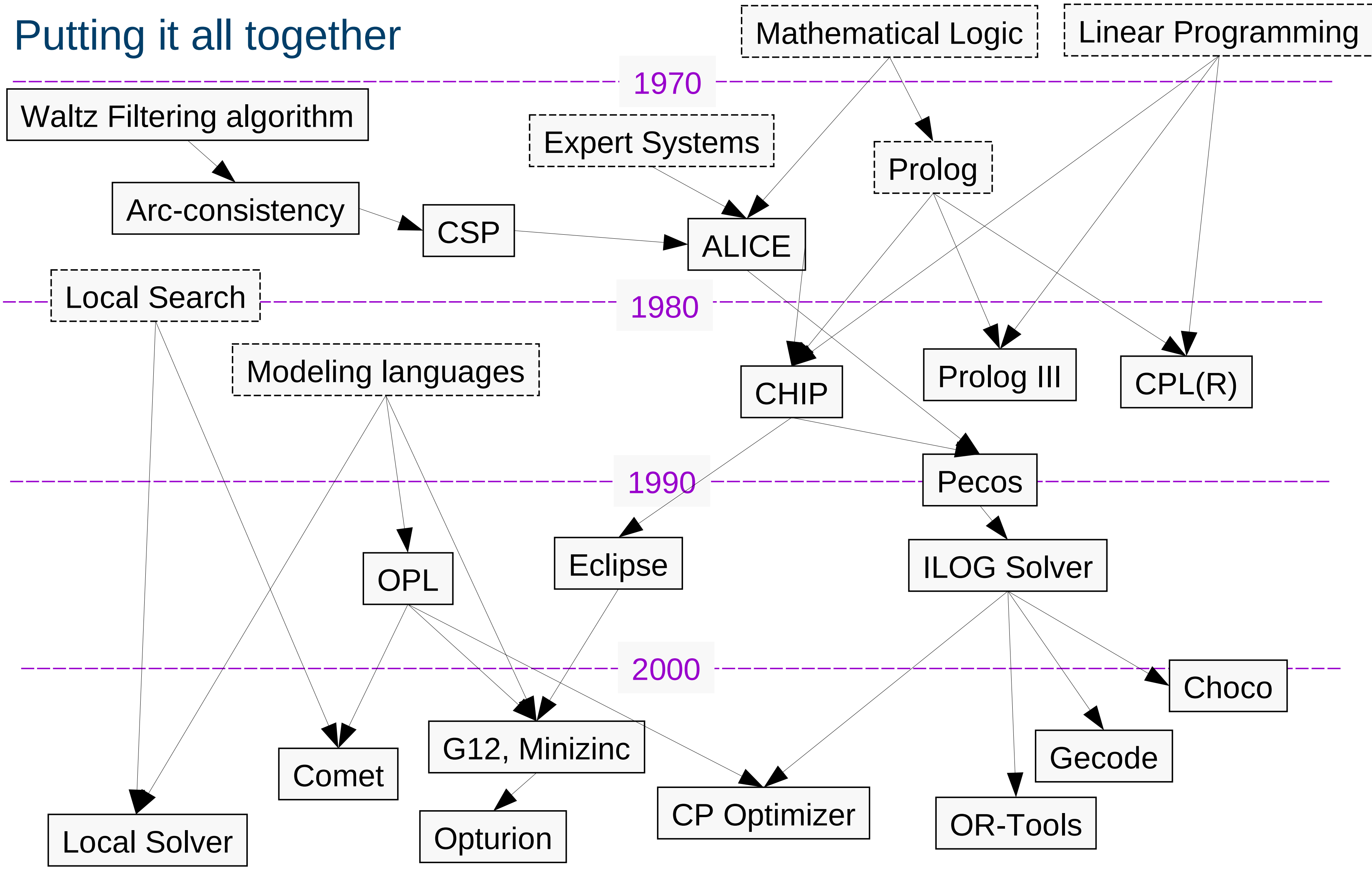
- Puget proposed that we go back to the “model-and-run” approach because CP was in a dead end
  - Not well received…(at the time)
- Progress on automatic search
  - *e.g*. Impact Search (Refalo 2004)
  - International workshops and sessions on “autonomous” search
- ILOG effort, resulted in CP Optimizer
  - First modern automatic CP solver
- Others use different approaches based on local search
  - *e.g.* COMET, LocalSolver

Most Recently

- CP is being integrated into modelling tools:
  - OPL (CP Optimizer)
  - AIMMS (CP Optimizer)
  - AMPL (CP Optimizer, Gecode)

- New solvers are still being built, for example:
  - Chuffed (G12 project - uses lazy clause generation)
  - Opturion (Commercial spin off from G12 project)
  - OR-tools (Google, started by ex-ILOG people)
  - LocalSolver (Local search over 0-1 variables)

Putting it all together



Constraint programming decoded

What Constraint Programming people call it	What Math Programming people should understand
Programming	Computer programming
Planning	Programming
Solution	Feasible solution
Optimal solution	Solution
Variable	Decision variable
Variable Domain	Variable bounds or set of admissible values
Constraints	Not limited to linear, quadratic or mixed integer
Tree Search	Branch & Bound
Heuristics	Branching strategy
Constraint inference	Presolve
Constraint Propagation	Bound strengthening
Global constraint	Specialized algorithm

Conclusion

- CP is an alternative to MIP solvers
  - And can be used in combination with them

- Good for combinatorial problems
  - Optimization too of course, but optimality proof has not been the focus

- Major sweet spot is scheduling. *e.g.* In CP Optimizer:
  - Precedence constraints, resources, reservoirs, optional tasks
  - Work breakdown hierarchies
  - Best known results on many benchmarks

- Solvers are generally free for academic use, even commercial solvers. Give them a go!